**Advanced C++ | Conversion Operators**

In C++, the programmer abstracts real world objects using classes as concrete types. Sometimes it is required to convert one concrete type to another concrete type or primitive type implicitly. Conversion operators play smart role in such situations.

For example consider the following class

|  |
| --- |
| #include <iostream>  #include <cmath>    using namespace std;    class Complex  {  private:      double real;      double imag;    public:      // Default constructor      Complex(double r = 0.0, double i = 0.0) : real(r), imag(i)      {}        // magnitude : usual function style      double mag()      {          return getMag();      }        // magnitude : conversion operator      operator double ()      {          return getMag();      }    private:      // class helper to get magnitude      double getMag()      {          return sqrt(real \* real + imag \* imag);      }  };    int main()  {      // a Complex object      Complex com(3.0, 4.0);        // print magnitude      cout << com.mag() << endl;      // same can be done like this      cout << com << endl;  } |

We are printing the magnitude of Complex object in two different ways.

Note that usage of such smart (over smart ?) techniques are discouraged. The compiler will have more control in calling an appropriate function based on type, rather than what the programmer expects. It will be good practice to use other techniques like class/object specific member function (or making use of C++ Variant class) to perform such conversions. At some places, for example in making compatible calls with existing C library, these are unavoidable.